

IN THE CLAIMS

1. (Currently Amended) A method, comprising:
 - analyzing source codes of a main thread having one or more delinquent loads, the one or more delinquent loads representing loads which likely suffer cache misses during an execution of the main thread, the source codes including one or more code regions, each code region corresponding to a sequence of instructions representing an iteration loop in the source codes, the one or more code regions sharing at least one instruction in the source codes;
 - estimating a communication cost between the main thread and each code region;
 - selecting a code region from the one or more code regions for one or more helper threads with respect to the main thread based on the analysiscommunication cost; and
 - generating codes for the one or more helper threads, the one or more helper threads being speculatively executed in parallel with the main thread to perform one or more tasks for the selected code region of the main thread.
2. (Currently Amended) The method of claim 1, wherein analyzing the source codes comprises:
 - generating one or more profiles for cache misses of the selected code region; and
 - analyzing the one or more profiles to identify one or more candidates for thread-based prefetch operations.
3. (Currently Amended) The method of claim 2, wherein generating one or more profiles comprises:
 - executing an application associated with the main thread with debug information; and

sampling cache misses and accumulating hardware counter for each static load of the selected code region to generate the one or more profiles for each cache hierarchy.

4. (Previously Presented) The method of claim 3, wherein analyzing the one or more profiles comprises:
 - correlating the one or more profiles with respective source code based on the debug information; and
 - identifying top loads that contribute cache misses above a predetermined level as the delinquent loads.
5. (Previously Presented) The method of claim 1, wherein analyzing the source codes comprises:
 - building a dependent graph that captures data and control dependencies of the main thread; and
 - performing slicing operations on the main thread based on the dependent graph to generate code slices, each code slice corresponding to one of the one or more delinquent loads.
6. (Currently Amended) The method of claim 5, wherein selecting the code region further comprises:
 - limiting traversal of the dependency graph to be within the selected code region for the slicing operations;
 - merging two or more of the code slices into a helper thread of the one or more helper threads to minimize code duplication;
 - computing liveness information providing communication cost between the main thread on the one of the helper threads;

determining a communication scheme communicating live-in values between the main thread and the helper thread according to the liveness information, wherein the live-in values are accessed in the helper thread without re-computation; and determining a change in size of the helper thread according to one of the slicing operations corresponding to a separate code region of the one or more overlapping code regions, wherein the selected code region encompasses the separate code region, wherein the change reduces the size of the helper thread.

7. (Previously Presented) The method of claim 6, wherein selecting the code region further comprises determining a synchronization period for the helper thread to synchronize the main thread and the helper thread, the helper thread performing its tasks within the synchronization period.

8. (Currently Amended) A machine-readable storage medium having executable code to cause a machine to perform a method, the method comprising:
- analyzing source codes of a main thread having one or more delinquent loads, the one or more delinquent loads representing loads which likely suffer cache misses during an execution of the main thread, the source codes including one or more code regions, each code region corresponding to a sequence of instructions representing an iteration loop in the source codes, the one or more code regions sharing at least one instruction in the source codes;
- selecting a code region from the one or more code regions for one or more helper threads with respect to the main thread based on the analysis; and
- generating software codes for the one or more helper threads, the one or more helper threads ~~being to be~~ speculatively executed in parallel with the main thread to perform one or more prefetching tasks for the selected code region of the main thread.

wherein the generated software codes include synchronization codes for the one or more helper threads to synchronize with the main thread during the execution.

9. (Currently Amended) The machine-readable storage medium of claim 8, wherein analyzing the source codes comprises:
 - generating one or more profiles for cache misses of the selected code region; and
 - analyzing the one or more profiles to identify one or more candidates for thread-based prefetch operations.
10. (Currently Amended) The machine-readable storage medium of claim 9, wherein generating one or more profiles comprises:
 - executing an application associated with the main thread with debug information; and
 - sampling cache misses and accumulating hardware counter for each static load of the selected code region to generate the one or more profiles for each cache hierarchy.
11. (Previously Presented) The machine-readable storage medium of claim 10, wherein analyzing the one or more profiles comprises:
 - correlating the one or more profiles with respective source code based on the debug information; and
 - identifying top loads that contribute cache misses above a predetermined level as the delinquent loads.
12. (Previously Presented) The machine-readable storage medium of claim 8, wherein analyzing the source codes comprises:
 - building a dependent graph that captures data and control dependencies of the main thread; and

performing slicing operations on the main thread based on the dependent graph to generate code slices, each code slice corresponding to one of the one or more delinquent loads.

13. (Currently Amended) The machine-readable storage medium of claim 12, wherein selecting the code region further comprises:

limiting traversal of the dependency graph to be within the selected code region for the slicing operations;

merging two or more of the code slices into a helper thread of the one or more helper threads to minimize code duplication;

computing liveness information providing communication cost between the main thread on the one of the helper threads;

determining a communication scheme communicating live-in values between the main thread and the helper thread according to the liveness information, wherein the live-in values are accessed in the helper thread without re-computation; and

determining a change in size of the helper thread according to one of the slicing operations corresponding to a separate code region of the one or more overlapping code regions, wherein the selected code region encompasses the separate code region, wherein the change reduces the size of the helper thread.

14. (Currently Amended) The machine-readable storage medium of claim ~~13~~8, wherein ~~generating the software codes wherein selecting the code region~~ further comprises determining a synchronization period ~~for the helper thread to synchronize the main thread and the helper thread~~for the software codes such that; the helper thread ~~performing~~is to perform its respective prefetching tasks within the synchronization period.

15. (Currently Amended) A data processing system, comprising:
a memory storing executable instructions including a compiler; and
a processor ~~capable of performing multi-threading operations~~coupled to the memory to
execute the executable instructions from the memory for performing
multi-threading operations, the processor being configured to;
~~a memory coupled to the processor; and~~
~~a process executed by the processor from the memory to cause the processor to~~
analyze source codes of a main thread having one or more delinquent loads,
the one or more delinquent loads representing loads which likely suffer
cache misses during an execution of the main thread, the source codes
including one or more code regions, each code region corresponding to
a sequence of instructions representing an iteration loop in the source
codes, the one or more code regions sharing at least one instruction in
the source codes,
estimate a communication cost between the main thread and each code region,
select the code region from the one or more code regions for one or more
helper threads with respect to the main thread based on the
~~analysis~~communication cost, and
generate code for the one or more helper threads, the one or more helper
threads being speculatively executed in parallel with the main
thread to perform one or more tasks for the selected code region of the
main thread.
16. (Original) The data processing system of claim 15, wherein the process is executed by a
compiler during a compilation of an application.
17. (Currently Amended) A method, comprising:

executing a main thread of an application in a multi-threading system; and
spawning one or more helper threads from the main thread created from
source codes including one or more code regions sharing at least one
instruction in the source codes, each code region associated with an estimation
of communication cost with the main thread, the code region corresponding
to a sequence of instructions representing an iteration loop in the source codes,
the one or more helper thread to perform one or more computations for the
main thread when the main thread enters a code region selected from the one or
more code regions based on the estimation of communication cost, the selected
code region having one or more delinquent loads, ~~each code region
corresponding to a sequence of instructions representing an iteration loop in the
source codes~~, the one or more helper threads being created separately from the
source codes of the main thread during a compilation of the source codes for
the main thread.

18. (Original) The method of claim 17, further comprising:

creating a thread pool to maintain a list of thread contexts; and
allocating one or more thread contexts from the thread pool to generate the one or
more helper threads.

19. (Previously Presented) The method of claim 18, further comprising:

terminating the one or more helper threads when the main thread exits the code region;
and
releasing the thread contexts associated with the one or more helper threads back to
the thread pool.

20. (Previously Presented) The method of claim 17, wherein the one or more help threads are placed in a run queue prior to execution, further comprising
determining a time period for each of the helper threads in the run queue, each of the
helper threads being terminated from the run queue when the respective time
period expires.
21. (Original) The method of claim 20, wherein each of the helper threads terminates when the time period expires even if the respective helper thread has not been accessed by the main thread.
22. (Previously Presented) The method of claim 17, further comprising discarding results generated by the one or more helper threads when the main thread exits the code region, the results not being reused by another code region of the main thread.
23. (Currently Amended) A machine-readable storage medium having executable code to cause a machine to perform a method, the method comprising:
executing a main thread of an application in a multi-threading system; and
spawning one or more helper threads from the main thread created from source codes
including one or more code regions sharing at least one instruction of the
source codes, each code region associated with an estimation of
communication cost with the main thread, the code region corresponding
to a sequence of instructions representing an iteration loop in the source codes,
the one or more helper threads to perform one or more computations for the
main thread when the main thread enters a code region selected from the one or
more code regions based on the estimation of communication cost, the selected
code region having one or more delinquent loads, ~~each code region
corresponding to a sequence of instructions representing an iteration loop in the~~

~~source codes,~~ the one or more helper thread being created separately from the source codes of the main thread during a compilation of the source codes for the main thread.

24. (Previously Presented) The machine-readable storage medium of claim 23, wherein the method further comprises:
- creating a thread pool to maintain a list of thread contexts; and
 - allocating one or more thread contexts from the thread pool to generate the one or more helper threads.
25. (Previously Presented) The machine-readable storage medium of claim 24, wherein the method further comprises:
- terminating the one or more helper threads when the main thread exits the code region;
 - and
 - releasing the thread contexts associated with the one or more helper threads back to the thread pool.
26. (Previously Presented) The machine-readable storage medium of claim 23, wherein the one or more help threads are placed in a run queue prior to execution, and wherein the method further comprises determining a time period for each of the helper threads in the run queue, each of the helper threads being terminated from the run queue when the respective time period expires.
27. (Previously Presented) The machine-readable storage medium of claim 26, wherein each of the helper threads terminates when the time period expires even if the respective helper thread has not been accessed by the main thread.

28. (Previously Presented) The machine-readable storage medium of claim 23, wherein the method further comprises discarding results generated by the one or more helper threads when the main thread exits the code region, the results not being reused by another code region of the main thread.
29. (Currently Amended) A data processing system, comprising:
~~a processor capable of performing multi-threading operations;~~
~~a memory coupled to the processor storing executable instructions for multi-threading operations;~~ and
~~a process executed by the processor coupled to the memory to execute the executable instructions from the memory, to cause the processor being configured to~~
execute a main thread of an application in a multi-threading system, and
spawn one or more helper threads from the main thread created from source codes including one or more code regions sharing at least one instruction of the source codes, each code region associated with an estimation of communication cost with the main thread, the code region corresponding to a sequence of instructions representing an iteration loop in the source codes, the one or more helper threads to
perform one or more computations for the main thread when the main thread enters a code region selected from the one or more code regions based on the estimation of communication cost, the selected code region having one or more delinquent loads, each code region
~~corresponding to a sequence of instructions representing an iteration loop of the source codes,~~ the one or more helper thread being created separately from the source codes of the main thread during a compilation of the source codes for the main thread.

30. (Canceled)